Secure Proximity-Based Identity Pairing using an Untrusted Signalling Service

Tim Panton
Westhawk Limited
TechHub Manchester
Manchester, UK
Email: thp@westhawk.co.uk

David Llewellyn-Jones*
Computer Laboratory
University of Cambridge
Cambridge, UK
Email: dl551@cam.ac.uk

Nathan Shone, Mahmoud Hashem Eiza
Department of Computer Science
Liverpool John Moores University
Liverpool, UK
Email: {N.Shone, M.Hashemeiza}@ljmu.ac.uk

***Abstract*—New protocols such as WebRTC promise seamless in-browser peer-to-peer communications that in theory remove the need for third-party services. In practice, widespread use of Firewalls, NATS and dynamic IP addresses mean that overlay addressing or use of some fixed rendezvous point is still needed. In this paper we describe a proximity-based pairing scheme that uses a signalling service to minimise the trust requirements on the third party, achieving anonymity and avoiding the need for PKI, while still requiring only a simple asymmetric pairing protocol.**

I. Introduction

Although the Internet was originally designed to allow seamless point-to-point communication using just IP addresses as a universal addressing scheme, the simplicity of this approach rarely applies in practice. Internet engineers introduced Network Address Translation (NAT) to allow multiple devices to share a single IP address. This also had the effect of restricting links to only those initiated through outbound connections. Ingress-filtering firewalls introduce similar restrictions, aiming to prevent externally initiated threats while minimising the restrictions imposed on those inside the firewall.

To a large extent the Web has been unaffected by this, given its firm adherence to a client-server design philosophy. However, as a result Web-based services tend to be highly centralised and users are therefore heavily reliant on trust in the organisations that run them.

Many current projects aim to offer a more federated and distributed approach to communications, attempting to avoid the need to rely on monolithic trusted intermediaries. Web-based examples such as diaspora* [4], ownCloud [15] and pump.io [19] offer federated Web-based alternatives to the likes of Facebook, Dropbox and Twitter respectively [16].

While these approaches focus on federation, there have also been attempts to introduce a peer-to-peer approach to the Web in general. In June 2009 Opera released Opera Unite, giving users the ability to publish their own Web-content directly from their browser, hosted locally. Effectively it integrated a Web server into the browser, using UPnP for local connections and an Opera-hosted proxy for NAT traversal. Opera Unite failed to gain traction however, and was disabled as a default feature of the Opera browser in June 2012 [14].

*Dr Llewellyn-Jones, now at Cambridge, was at Liverpool John Moores University, Dept. of Computer Science, while this research was carried out.

One of the most recent and successful attempts to introduce direct browser-to-browser communications has been WebRTC (Web Real Time Communications) protocol. Currently a draft W3C standard [3], WebRTC is already being deployed in Firefox, Chrome, Opera and Edge. WebRTC uses the Interactive Connectivity Establishment (ICE) protocol [21], which uses other techniques such as Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN), to negotiate NAT boundaries. We describe the WebRTC protocol more fully in the next section.

WebRTC can be a fully point-to-point protocol between Web browsers. However, the specification does not include any means of determining or authenticating endpoints; it assumes this step has already taken place. As a result, there is a need for a method of securely determining the end points for a communication session.

In this paper we present such a method using a proximity-based identity pairing mechanism. There are three important elements to this. The first is the use of unidirectional proximity pairing (using a QR code) as a means of authentication for tying users to devices. The second is the use of an untrusted signalling service for matching devices to IP addresses. The third is the combining of certificate fingerprints with identities. The novelty of this work derives from the dual use of fingerprint and ID, as a means of providing anonymous communications via an untrusted signalling server, which also provides robust authentication for the end-points without the need for any pre-trusted architecture such as Public Key Infrastructure (PKI).

II. The WebRTC Protocol

Although still only at draft W3C stage, the fact of its integration into the most popular Web browsers has set up WebRTC as a *de facto* standard for real-time communications. In theory the media format is unrestricted, but in practice WebRTC is primarily used for video and voice calls between pairs of individuals. Google Hangouts, amongst many other services, already makes use of WebRTC.

WebRTC builds on numerous other standards such as ICE for NAT traversal, Datagram Transport Layer Security (DTLS) for establishing a secure session, Secure Real-time Transport Protocol (SRTP) for real-time streaming and Secure Real Time

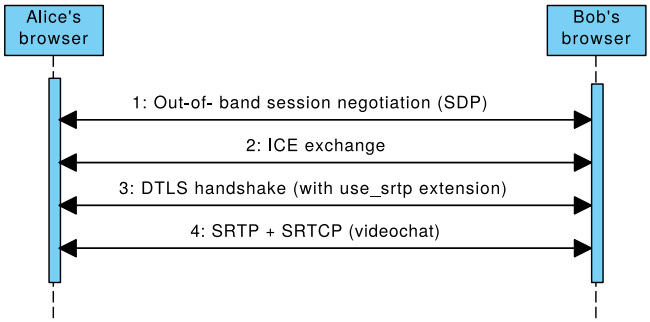

Fig. 1: Overview of WebRTC protocol

Control Protocol (SRTCP) for protecting control messages. DTLS-SRTP is a key exchange mechanism that is made mandatory in WebRTC where DTLS is used to establish keying material and algorithms for the SRTP session. SRTP then protects the application data sent over the WebRTC data channels. Figure 1 provides a high-level overview of how WebRTC leverage existing standards. Although WebRTC defines Javascript methods for generating the offer and answer messages used to negotiate the session, the means of relaying these between browsers is left unspecified. The SRTP communication is protected using a key derived from the master secret key established during the DTLS stage.

DTLS will commonly use a trust chain to authenticate the server's certificate, relying on the established PKI for this. However for client-client communications, WebRTC certificates are generated on-the-fly. Consequently it's infeasible to have them signed by an infrastructure CA. The WebRTC draft does not propose any mechanism for validating these certificates. It also defines no signalling channel for initiating a session. However, the WebRTC defines the WRTCoffer and WRTCanswer data structures that are assumed to be shared between two clients as a precursor to communication, shown as step 1 in Figure 1 during the session negotiation process. The session negotiation process is carried out using the Session Description Protocol (SDP). The WRTCoffer data type can include a certificate fingerprint field, which can be SHA-128 or SHA-256 in some browsers (*e.g.* Chrome). The WebRTC standard itself does not specify the hash function type. We utilise this field as a critical part of our proposed protocol. Without PKI or other validation mechanisms WebRTC would be vulnerable to a man-in-the-middle (MitM) attack, since the end-points do not know the certificates in advance.

We prevent the attack by pre-sharing the fingerprints using the Westhawk protocol defined in Section IV. Any alteration to the certificates can then be immediately identified as shown in Section VI, allowing any MitM attack to be identified and the session aborted.

III. Literature Review

As the WebRTC standards are still in the draft stage [3], there are currently no definitive authentication requirements. Therefore, various solutions are being considered for WebRTC, presently the most popular is DTLS-SRTP [9]. This uses the DTLS handshake to establish keying material, algorithms and parameters to secure RTP packets (becoming SRTP) [8]. There are both non-PKI and PKI versions of DTLS-SRTP but it depends upon a PKI to prevent MiTM attacks.

An alternative method being considered is ZRTP [18], which uses the Diffie-Hellman key exchange during the call setup to generate a shared secret, used to generate the keys and salt for the SRTP session. However, this does not protect against MiTM attacks, so to ensure the attacker is not present in the first encounter (*i.e.* when no shared secrets exist), ZRTP uses the short authentication string method, which involves both parties verbally cross-checking a shared value displayed at both end-points. Unlike DTLS-SRTP, ZRTP does not depend on a PKI to prevent MiTM attacks but is not considered as secure as a PKI version of DTLS-SRTP.

Bell Labs have also proposed a secure and lightweight authentication mechanism that could be used within WebRTC [8]. It involves the use of an extra key in the SIP setup, and the evaluation of a pseudo-random permutation (e.g. AES). Their protocol claims to be a trade-off between security, costs and deployment complexity. Therefore bridging the gap between the complexity of DTLS-SRTP and ZRTP, and the insecurity of Session Description Protocol Security Descriptions (SDES). Coincidently, SDES has largely been dropped from WebRTC due to its inadequate security capabilities [11].

The other main proposed method of authentication for WebRTC involves the use of Web-based identity management providers (IdM). The WebRTC draft standards propose the use of IdMs [9] but no mechanisms are specified [2]. The idea is that two parties can circumvent trusted signalling servers, by authenticating against a mutually-trusted third party (*e.g.* social networks), which can be accomplished browser-to-browser. The main solutions being considered include OpenID [1], oAuth [7] and BrowserID [2].

As WebRTC is entirely browser-based, it is important that authentication schemes are widely compatible, particularly with computationally-limited devices. It is for this reason that we have also examined the latest IoT, proximity-based and out-of-band authentication research.

An emerging trend in authentication is the use of mobile devices. For example, Sun *et al.* [24] propose a novel two-factor multi-touch authentication scheme in which users draw on a mobile touch-screen with one or multiple fingers. Users are authenticated based on the geometric properties of the drawn shape as well as the physiological characteristics (*e.g.* velocity, acceleration, pressure and finger size). Similarly, Li *et al.* [12] propose a near field proximity authentication scheme for smartphones. This scheme involves the user simultaneously drawing a zigzag pattern on the screens of two mobile phones placed next to each other, using two fingers from the same hand. The extracted datasets have many features (*e.g.* curvature, curvature distance, moving time), which are used to generate a cryptographic key using a private set intersection protocol. Additionally, Lu *et al.* [13] propose a two-factor mutual authentication scheme for smart-living devices, with a session key agreement mechanism. It uses the

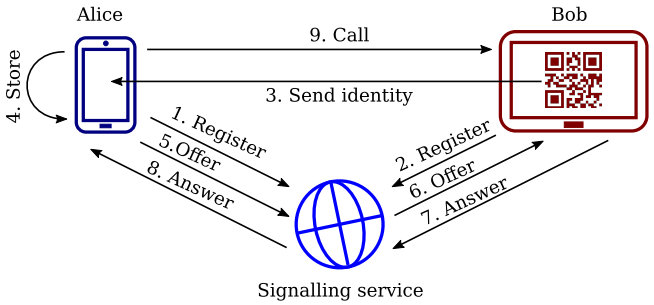

Fig. 2: Interactions between devices when pairing

devices' international mobile subscriber identities (IMSIs) and an identity-based remote mutual authentication scheme based on elliptic curve cryptography (ECC).

Another trending focus is ensuring authentication schemes remain lightweight, such as the work by Petrov *et al.* [20], which presents a many-to-many authentication scheme, based on passive NFC tags which store encrypted passphrases for each service used. The scheme ensures the passphrases can only be decrypted by the service for which it was intended to be used with. Another example is by Shivrav *et al.* [22], who propose a two-factor One Time Password (OTP) authentication scheme for IoT. It uses a lightweight identity-based elliptic curve cryptography scheme, which does not require key storage or depend on previous keys.

Finally, another trending focus is to attempt to integrate forms of user identity assurance into authentication schemes. For example, Shone *et al.* [23] present a two-factor mobile IoT device authentication scheme, one factor uses challenges derived from users' own digital memories. Durmus *et al.* [6] propose EAP-SocTLS, which is a decentralized approach for Wi-Fi access point authentication using embedded trust relations. It uses social networks to identify the trust relations between people and devices, and it claims to provide a more secure mechanism. Although this is for Wi-Fi, the concept of combining spatio-temporal and social network authentication could easily be applied to WebRTC.

While there are a large number of generalised pairing protocols to be found in the literature [17], in contrast the protocol presented here focusses on the combined use of certificate fingerprints and identities as a means of avoiding the need for a trusted intermediary. By integrating this into the proximity pairing protocol described in the next section to bootstrap the WebRTC session, we are able to replace the identity provider with an untrusted signalling service, thereby increasing privacy without compromising mutual authentication.

IV. Pairing Protocol

The Westhawk protocol involves a process of pairing two devices, followed by the establishment of a call using WebRTC. The WebRTC portion is largely unchanged, apart from the addition of some new authentication checks that can be performed locally and individually by the two devices (hence not altering the protocol itself). In this section we'll concentrate on the pairing protocol, which therefore represents the main contribution of this work.

We assume the use of two devices, one of which has camera capabilities (*e.g.* a phone) and the other has a screen (*e.g.* a tablet). Let's suppose Alice is using the phone and Bob the tablet, as shown in Figure 2.

The broad stages of the pairing protocol are that the two devices register their identities with a signalling service. These identities are generated as the hashes of the certificates (essentially the public keys) that will be used by the WebRTC protocol. Since they are not dependent on one another, the certificates can be generated in advance.

The phone and tablet are both able to connect via the Internet to the signalling service, and the service itself has been chosen in advance. However, Alice and Bob don't necessarily trust the signalling service.

Bob concatenates his identity (the same certificate hash) with a challenge nonce, which he then transfers to Alice using an out-of-band channel. In our case we use a QR code for this: Bob encodes his identity and the nonce in the QR code and displays it on screen. Alice captures a photograph of the screen and decodes it to reveal Bob's identity.

Alice can then send her WRTCoffer message to the signalling service, which the signalling service then passes on to Bob. Bob replies with an WRTCanswer message, sent via the signalling service to Alice.

At this stage both Alice and Bob then have the information they need to continue the standard WebRTC protocol without the need for the signalling service (although a separate intermediary may potentially be needed for the ICE exchange, but which we omit here for brevity).

The sequence diagram for this process is shown in Figure 3. From the user's point of view, both Alice and Bob press a 'pairing' button on their device (or an app or webpage button). Bob's device will then display a QR code that Alice scans, which from their point of view will complete the process.

Having done this, at any time in the future, either Alice or Bob can then initiate a call to the other without needing to repeat the process.

V. Implementation

An implementation of the pairing protocol has been deployed by the YoPet website [25], a service that allows owners to communicate with their pets while away from home. Figure 4 shows the pairing process using the QR code to transmit the ID of a laptop B to a phone A. The laptop is then positioned to allow Alice's pet (who happens to be called Bob) to view the screen and be seen by the laptop camera.

Having established this pairing, Alice can then remotely activate her laptop while away from home using her phone to 'videochat' with her pet, as shown in Figure 5.

VI. Security Analysis

Numerous potential security vulnerabilities could manifest themselves in a protocol along these lines. Hence, the following sections consider such cases and discuss the potential strengths and weaknesses of the protocol in relation to them.

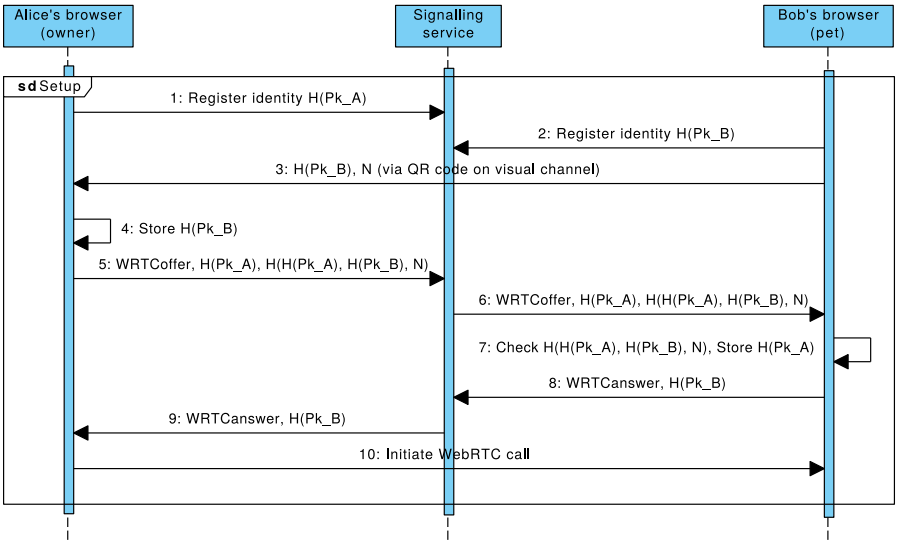

Fig. 3: Protocol sequence diagram using asymmetric pairing

We premise this discussion on an active attack model (informally the Dolev-Yao security model [5]), with the exception of the QR code visual channel. The proposed approach assumes this step to be taken by a human bringing together two trusted devices. While we discuss some possible attacks related to this in the text below, our default assumption is that this provides a secure channel (no interception, no chance of alteration, no spoofing), in contrast to the insecure network.

A. *Mutual Authentication*

Authentication is achieved by transferring identities between the two devices. As noted above, these identities fulfil a dual purpose. First they represent the means for the two devices to bootstrap the call routing process via the signalling service. Second they represent the means for verifying each others' identities, since they are generated as the hash of the public keys used to encrypt and authenticate. The associated private keys never leave their respective devices.

The authentication process from Bob to Alice is therefore straightforward, since it's achieved by transferring the QR code through the visual channel (step 3 in Figure 3). At this point B's identity isn't secret (it's sent in the clear at step 2). However, the only way for an attacker to capitalise on this would be to present B's identity as its own in step 3, which would require the compromise of one of the devices, the browser or the Javascript served to the browser. While any of these would render the protocol insecure, our assumption for the visual channel, using two trusted devices, leaves this outside the scope of our threat model.

The reverse authentication is less straightforward, since this is sent over the Internet in step 5 without the benefit of a secure proximity-based channel. We must check that if a MitM were to intercept and alter this message, or replay a previous message, that Bob would be able to identify this. This is particularly important at this stage given the signalling service is still acting as a forwarder, so is in an ideal position to perform such a MitM attack.

The integrity of the message is achieved through the use of the challenge nonce N transferred from B to A in step 3, and used to generate the hash value $H(H(Pk_A), H(Pk_B),$

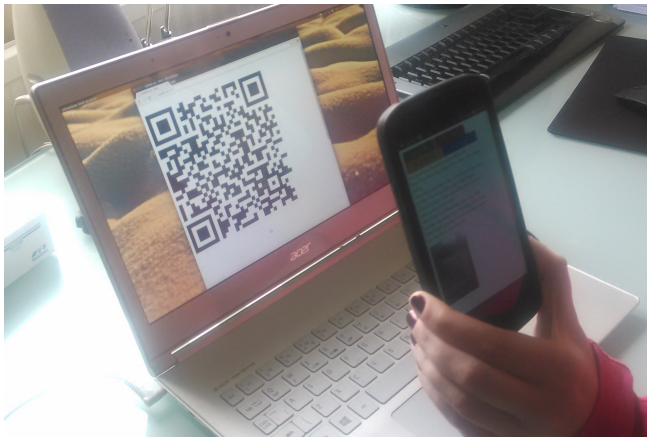

Fig. 4: YoPet device pairing using visual channel

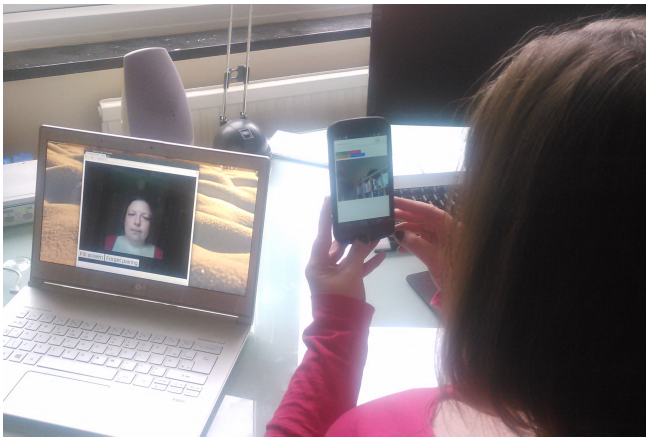

Fig. 5: YoPet video communication using WebRTC

```
{"type":"answer","sdp":"
v=0
o=mozilla...THIS_IS_SDPARTA-41.0.1 4104634441...
s=-
t=0 0
a=sendrecv
a=fingerprint:sha-256 BF:3E:50:AE:15:FF:5E:D8...
a=group:BUNDLE sdparta_0
a=ice-options:trickle
..."}
```

Fig. 6: Example WebRTC answer structure

N) transmitted in step 5. An attacker is unable to generate this hash, or a version containing any changes, since they will be unaware of the value N. Similarly, given that N is freshly generated for each run of the protocol, any attempt to replay an old message will be immediately identifiable by B on regenerating the hash locally with the correct N (step 7).

It's worth noting that as long as N is sufficiently random, neither A nor B need to remember previous values, since the transfer of N from B to A in step 3 cannot be intercepted, altered or replayed (by virtue of the channel used to perform the transfer). However, it is crucial that the values of N are unpredictable and with sufficiently high entropy to ensure an attacker, replaying an old message at step 6, will chance upon the same value N with sufficiently low probability.

Having performed the check at step 7, B can be sure that the identity $H(Pk_A)$ is indeed that of A, thereby achieving the authentication result. This then allows the certificates used during the call itself to be authenticated, by comparing their hashes against these identity values.

B. Man in the Middle Attacks

In the previous section we discussed the potential for a MitM attack to subvert the authentication process, and saw why an attempt up to and including step 7 wouldn't be successful. In steps 8 and 9 the WebRTC answer is returned from B to A in the form of an SDP response. Although this structure includes B's identity, this isn't tied cryptographically to the rest of the data. As such, it's important to be aware that the integrity of the answer can't be checked directly. The connection between Alice/Bob and the Signalling Service may be TLS-encrypted, but with the Signalling Service as an intermediary this doesn't represent end-to-end encryption.

The WebRTC answer contains details of the audio and video streams that will be transmitted in SDP format wrapped in a JSON structure, an excerpt of which is shown in Figure 6.

In practice, were a MitM attacker to change the contents of the SDP answer, this would cause the session to fail. Moreover, the fingerprint sent in the answer was already received by A in step 3, so this doesn't afford an attacker the opportunity to perform an impersonation attack, or to compromise the security of the later DTLS-SRTP session.

C. Visual Channel Attacks

A particular challenge when using visual channels such as QR codes is the danger of a replay attack [10], whereby an attacker captures an image of the QR code and displays this on a device elsewhere. The danger increases if the authentication mechanism is used for multiple services, since the QR code itself gives no indication about which service the user is logging in to. As a malicious service provider, I can therefore switch the QR code for my service with the QR code of some other service the user has an account with during the login process (potentially replaying it in realtime). The user thinks they're logging in to my service, but in reality are allowing *me* to log in to the user's account with the other service.

The equivalent in the case of our protocol would be to have Bob's browser display the QR code of some other service at step 3. The fact that Bob's device is trusted mitigates the risk compared to the classic case. Since the QR code is generated by the Javascript loaded in the browser, the attack requires the attacker to have control of this Javascript. This could happen in a number of ways: the device/browser could be compromised, the serving site could be compromised [1], or a phishing attack might cause the user to go to the wrong site.

These fall outside our attacker model, since we assume the device – including the browser and Javascript – to be trusted. Nevertheless, approaches for mitigating these attacks might include careful sandboxing and user care (*e.g.* checking the serving site's TLS certificate in the case of phishing).

Another related attack on the visual channel would be for an attacker to capture an image of the QR code displayed by B at step 3 and use it to pair with A before B is able to complete the process. Using this technique (*e.g.* by capturing the QR code using a long-distance camera) would provide the attacker the identity $H(Pk_A)$ and challenge N needed to complete the pairing process without suspicion.

The attack effectively challenges our assumption that the out-of-band visual channel is secure. Careful pairing by the user can reduce the likelihood of this (*e.g.* by avoiding pairing in public), as long as they are aware of the possibility.

D. Signalling Server Trust

One of the attractive features of WebRTC is it's peer-to-peer (browser-to-browser) nature, minimising the need to rely on intermediate services. Depending on the network structure, once the WebRTC SRTP connection has been established, the call can potentially be routed without the need to go through a centralised service. Our intention has been to retain this for the pairing protocol. However, the lack of universal addressing for browser-to-browser communication (*e.g.* due to the use of NATs) means that a rendezvous point of some sort is needed. This is provided by the signalling service.

Apart from the step 3 sent on the visual channel and the WebRTC call itself, all other pairing messages are sent through the Signalling Server, S. A natural question is therefore "to what extent must S be trusted". Ideally there should be no trust requirement, but this deserves closer inspection.

There are no secrets shared between A and S or B and S. The Signalling Service takes the same role as a potentially

[1] The required Javascript could be served locally to avoid this.

malicious MitM attacker and therefore introduces identical risks to those discussed in Section VI-B. It can alter the contents of messages which will result in the termination of the protocol. It can view the SDP offer/answer structures which are sent in the clear.

The identities $H(Pk_A)$ and $H(Pk_B)$ must be generated fresh for each pairing, since separate certificates will be used in each case, reducing the ability of S to correlate device pairings. However, fixed IP addresses could potentially be used to correlate device pairings. It has no opportunity to correlate calls since these are conducted peer-to-peer.

In practice then, S has minimal opportunity to compromise the process, having no more knowledge or access than any other potential MitM attacker.

VII. Conclusion

WebRTC has the potential to drive the development of many new browser-based realtime communication applications. While direct browser-to-browser communications and security are deeply embedded in the standard, the initial pairing process has been left intentionally unspecified.

While existing pairing mechanisms invariably rely on trusted intermediaries, such as identity providers, we remove this requirement using a novel repurposing of the certificate fingerprints as transient identity tokens. These can then be used by an untrusted signalling service to initiate a pairing. Moreover, we make use of a visual QR code to transfer the identity and challenge data needed to bootstrap the authentication process, particularly applicable to scenarios using trusted devices that minimise the potential for replay attacks. The handover to WebRTC that allows direct browser-to-browser streaming prevents the signalling service from tracking or associating future communication sessions.

Our security analysis demonstrates that the reliance on a trusted identity provider can be safely removed through the use of an untrusted signalling service combined with a careful pairing mechanism. Integrated with the direct communications supported by WebRTC our system provides particularly attractive properties from a privacy perspective.

The current approach assumes the use of trusted devices and a secure visual communications channel. In future work we intend to consider other mechanisms that may allow for more remote pairing. We also hope to consider the use of the protocol in the context of the Internet of Things, especially in scenarios where scope for visual input/output may be more limited. We are also interested in the potential for the security properties of the protocol to be validated formally.

Acknowledgment

The authors would like to thank Innovate UK for supporting the development of this paper.

References

[1] R. Barnes and M. Thomson. Browser-to-browser security assurances for WebRTC. *IEEE Internet Computing*, 18(6):11–17, 2014.

[2] V. Beltran, E. Bertin, and N. Crespi. User identity for WebRTC services: A matter of trust. *IEEE Internet Computing*, 18(6):18–25, 2014.

[3] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan. WebRTC 1.0: Real-time communication between browsers. http://www.w3.org/TR/webrtc/, Feb 2015.

[4] A. Bleicher. The anti-facebook. *IEEE Spectrum*, 48(6):54–82, Jun 2011.

[5] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, Mar 1983.

[6] Y. Durmus and K. Langendoen. WiFi authentication through social networks - a decentralized and context-aware approach. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 532–538. IEEE, March 2014.

[7] L. L. Fernandez, M. Gallego, B. Garcia, D. Fernández-López, and F. J. Lopez. Authentication, authorization, and accounting in WebRTC PaaS infrastructures: The case of Kurento. *IEEE Internet Computing*, 18(6):34–40, 2014.

[8] V. K. Gurbani and V. Kolesnikov. A secure and lightweight scheme for media keying in the session initiation protocol (SIP): Work in progress. In *Principles, Systems and Applications of IP Telecommunications*, IPTComm '10, pages 32–41, New York, NY, USA, 2010. ACM.

[9] IETF. Real-time communication in WEB-browsers. http://datatracker.ietf.org/wg/rtcweb/.

[10] G. Jenkinson, M. Spencer, C. Warrington, and F. Stajano. *Security Protocols XXII*, chapter I Bought a New Security Token and All I Got Was This Lousy Phish – Relay Attacks on Visual Code Authentication Schemes, pages 197–215. Lecture Notes in Computer Science. Springer International Publishing, Mar 2014.

[11] H. Kaplan. Should we support SDES in WebRTC? Technical report, IETF, Vancouver, BC, Canada, 2015., 2015.

[12] L. Li, X. Zhao, and G. Xue. A proximity authentication system for smartphones. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2015.

[13] Y.-F. Lu, I.-C. Shu, H.-W. Tseng, and S.-C. Chou. An NFC-phone mutual authentication scheme for smart-living applications. In *2014 International Conference on Information Science, Electronics and Electrical Engineering (ISEEE)*, volume 2, pages 1053–1057. IEEE, April 2014.

[14] J. Lyle, C. Nilsson, A. Isberg, and S. Faily. Extending the web to support personal network services. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC 13, pages 711–716. ACM, 2013.

[15] M. Mahjoub, A. Mdhaffar, R. Halima, and M. Jmaiel. A comparative study of the current cloud computing technologies and offers. In *2011 First International Symposium on Network Cloud Computing and Applications (NCCA)*, pages 131–134. IEEE, Nov 2011.

[16] W. Miltenburg. *Functional breakdown of decentralised social networks*. PhD thesis, University of Amsterdam, July 2015.

[17] S. Mirzadeh, H. Cruickshank, and R. Tafazolli. Secure Device Pairing: A Survey. *IEEE Communications Surveys Tutorials*, 16(1):17–40, 2014.

[18] P. Zimmerman, A. Johnston, and J. Callas. ZRTP: Media path key agreement for unicast secure RTP. https://tools.ietf.org/html/rfc6189.

[19] A. Pathak, G. Rosca, V. Issarny, M. Decat, and B. Lagaisse. *Engineering Secure Future Internet Services and Systems*, chapter Privacy and Access Control in Federated Social Networks, pages 160–179. Lecture Notes in Computer Science. Springer International Publishing, 2014.

[20] V. Petrov, S. Edelev, M. Komar, and Y. Koucheryavy. Towards the era of wireless keys: How the IoT can change authentication paradigm. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 51–56. IEEE, March 2014.

[21] J. Rosenberg. RFC 5245: Interactive connectivity establishment (ICE), Apr 2010.

[22] V. Shivraj, M. Rajan, M. Singh, and P. Balamuralidhar. One time password authentication scheme based on elliptic curves for internet of things (IoT). In *Information Technology: Towards New Smart World (NSITNSW), 2015 5th National Symposium on*, pages 1–6, Feb 2015.

[23] N. Shone, C. Dobbins, W. Hurst, and Q. Shi. Digital memories based mobile user authentication for IoT. In *Dependable, Autonomic and Secure Computing (DASC15), 13th IEEE International Conference on*, page PP. IEEE, 2015.

[24] J. Sun, R. Zhang, J. Zhang, and Y. Zhang. Touchin: Sightless two-factor authentication on multi-touch mobile devices. In *IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014*, pages 436–444, 2014.

[25] YoPet Team. Yopet - a new way to talk with your pet. https://yopet.us. Accessed 21/05/2015.